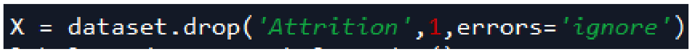
Data Preprocessing:

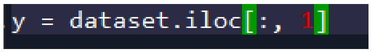
Before performing any modeling, the data must preprocess. Firstly, the independent and dependent variables are created.

Creation of the Independent variable (X).



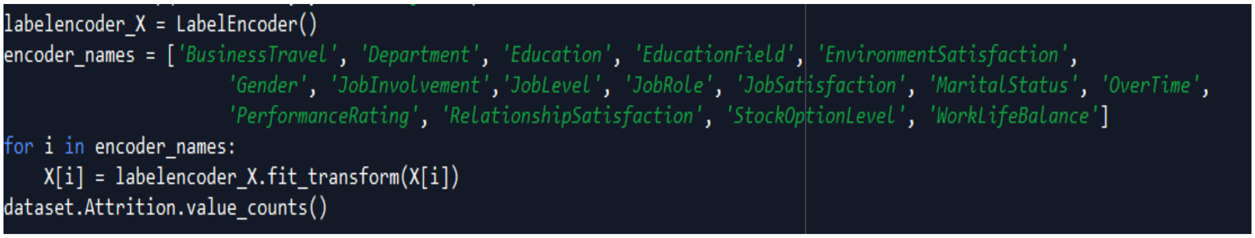
```
X = dataset.drop('Attrition',1,errors='ignore')
```

Creation of dependent variable(y).



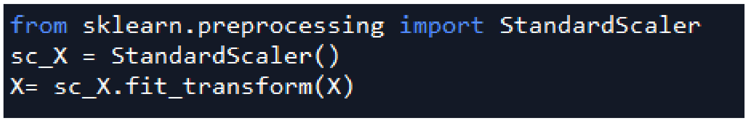
```
y = dataset.iloc[:, 1]
```

A label encoder is applied to X to deal with the categorical data in it.



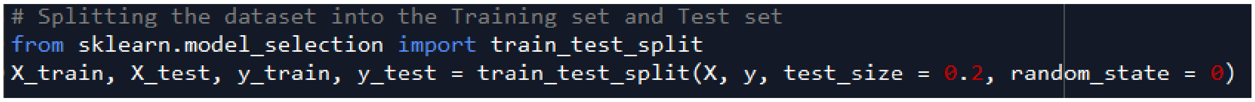
```
labelencoder_X = LabelEncoder()
encoder_names = ['BusinessTravel', 'Department', 'Education', 'EducationField', 'EnvironmentSatisfaction',
                 'Gender', 'JobInvolvement','JobLevel', 'JobRole', 'JobSatisfaction', 'MaritalStatus', 'OverTime',
                 'PerformanceRating', 'RelationshipSatisfaction', 'StockOptionLevel', 'WorkLifeBalance']
for i in encoder_names:
    X[i] = labelencoder_X.fit_transform(X[i])
dataset.Attrition.value_counts()
```

Perform feature scaling on the Independent variable.



```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X= sc_X.fit_transform(X)
```

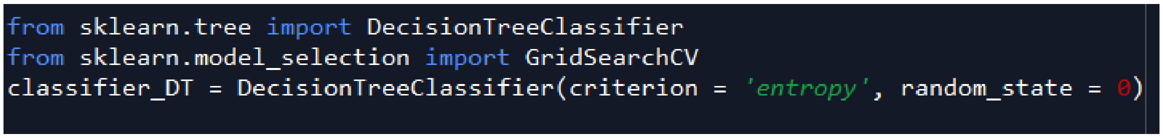
Split the X and y variable into training and test set



```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

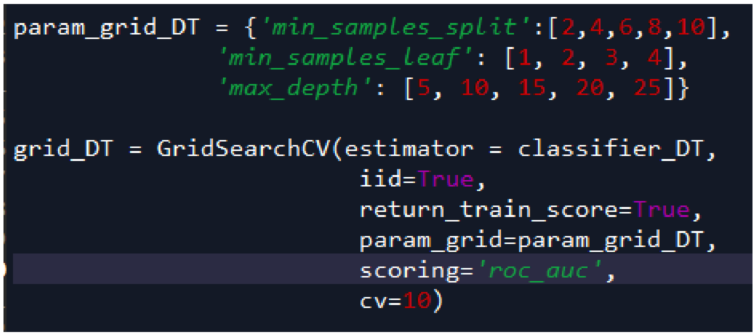
Decision Tree:

The libraries are imported.



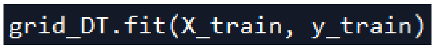
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
classifier_DT = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
```

The model is optimized using grid search.



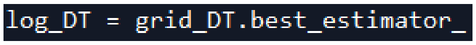
```
param_grid_DT = {'min_samples_split':[2,4,6,8,10],
                 'min_samples_leaf': [1, 2, 3, 4],
                 'max_depth': [5, 10, 15, 20, 25]}

grid_DT = GridSearchCV(estimator = classifier_DT,
                       iid=True,
                       return_train_score=True,
                       param_grid=param_grid_DT,
                       scoring='roc_auc',
                       cv=10)
```

The optimized model is fitted on the training set of the X and y variables.



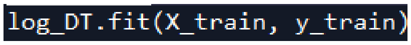
```
grid_DT.fit(X_train, y_train)
```

The best estimator instance is used on the fitted optimized model



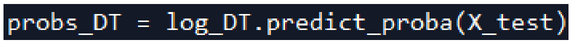
```
log_DT = grid_DT.best_estimator_
```

The best estimator model is then fitted to the training set of X and y.



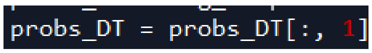
```
log_DT.fit(X_train, y_train)
```

The test set of X is the predicted. This will produce a result showing the probabilities of both an employee leaving or not.



```
probs_DT = log_DT.predict_proba(X_test)
```

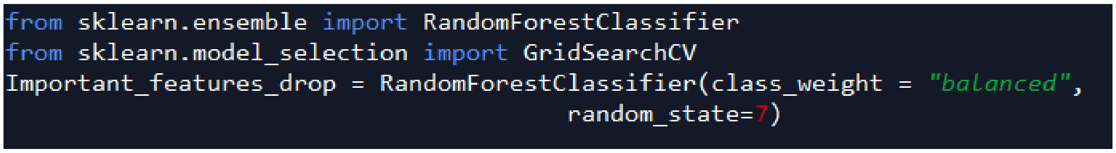
But we want the probabilities of an employee leaving.



```
probs_DT = probs_DT[:, 1]
```

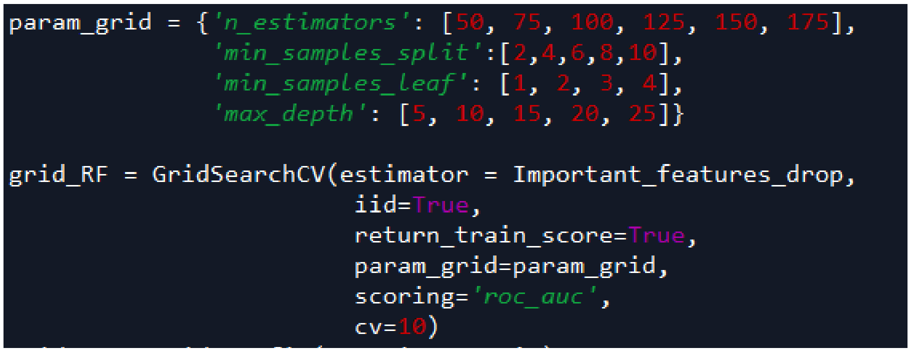
Random Forest regression:

The necessary libraries are first imported.



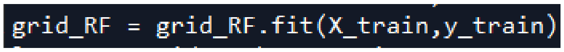
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
Important_features_drop = RandomForestClassifier(class_weight = "balanced",
                                                 random_state=7)
```

The model is optimized using grid search.



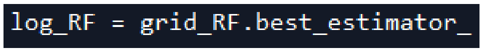
```
param_grid = {'n_estimators': [50, 75, 100, 125, 150, 175],
              'min_samples_split':[2,4,6,8,10],
              'min_samples_leaf': [1, 2, 3, 4],
              'max_depth': [5, 10, 15, 20, 25]}

grid_RF = GridSearchCV(estimator = Important_features_drop,
                        iid=True,
                        return_train_score=True,
                        param_grid=param_grid,
                        scoring='roc_auc',
                        cv=10)
```

The optimized model is fitted on the training set of the X and y variables.



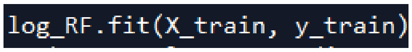
```
grid_RF = grid_RF.fit(X_train,y_train)
```

The best estimator instance is used on the fitted optimized model



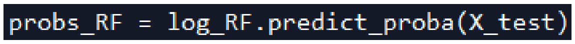
```
log_RF = grid_RF.best_estimator_
```

The best estimator model is then fitted to the training set of X and y



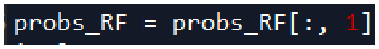
```
log_RF.fit(X_train, y_train)
```

The test set of X is the predicted. This will produce a result showing the probabilities of both an employee leaving or not.



```
probs_RF = log_RF.predict_proba(X_test)
```

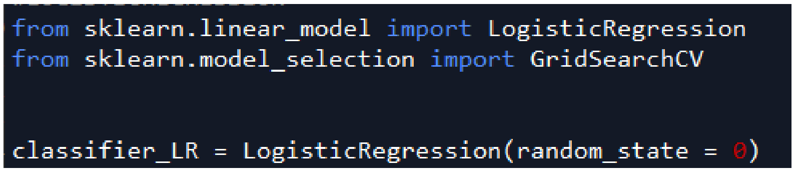
But we want the probabilities of an employee leaving.



```
probs_RF = probs_RF[:, 1]
```

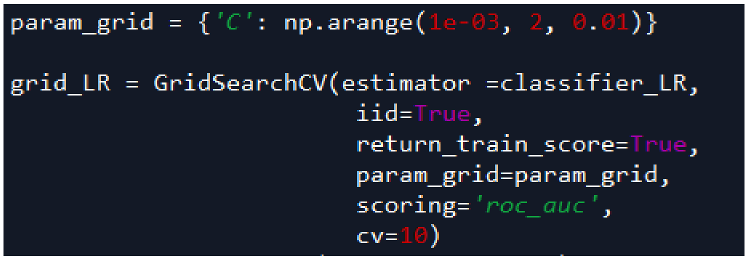
Logistic Regression:

The necessary libraries are imported.



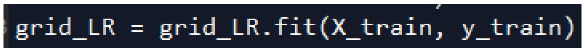
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV


classifier_LR = LogisticRegression(random_state = 0)
```

The model is optimized using grid search.



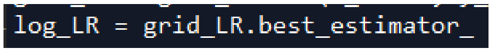
```
param_grid = {'C': np.arange(1e-03, 2, 0.01)}

grid_LR = GridSearchCV(estimator =classifier_LR,
                       iid=True,
                       return_train_score=True,
                       param_grid=param_grid,
                       scoring='roc_auc',
                       cv=10)
```

The optimized model is fitted on the training set of the X and y variables.



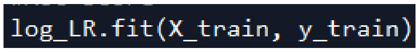
```
grid_LR = grid_LR.fit(X_train, y_train)
```

The best estimator instance is used on the fitted optimized model



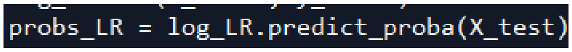
```
log_LR = grid_LR.best_estimator_
```

The best estimator model is then fitted to the training set of X and y. This will produce a result showing the probabilities of both an employee leaving or not.



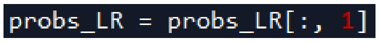
```
log_LR.fit(X_train, y_train)
```

The test set of X is the predicted. This will produce a result showing the probabilities of both an employee leaving or not.



```
probs_LR = log_LR.predict_proba(X_test)
```

But we want the probabilities of an employee leaving.

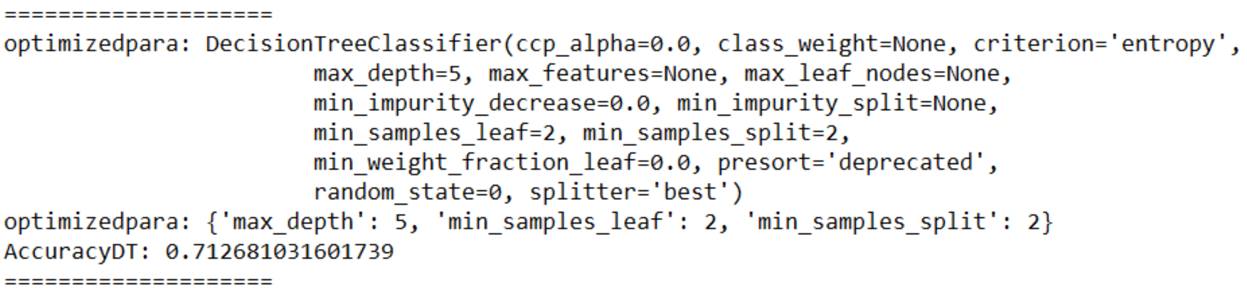


```
probs_LR = probs_LR[:, 1]
```

Best Model:

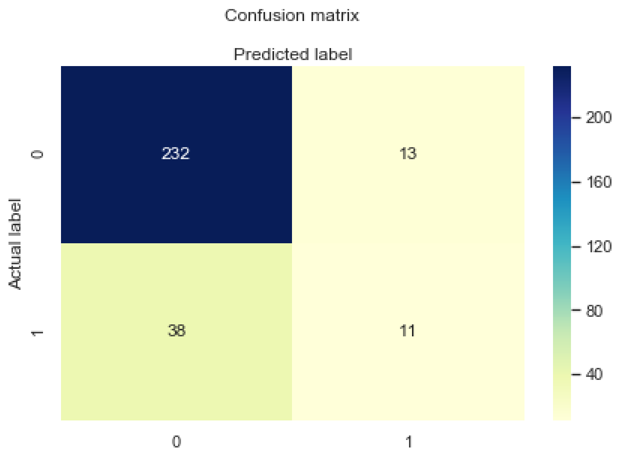
To find the best model of the three, we must do some analysis. Three techniques were used to determine which was the best model of all, there are; Accuracy, Confusion matrix and ROC curve.

Decision Tree:



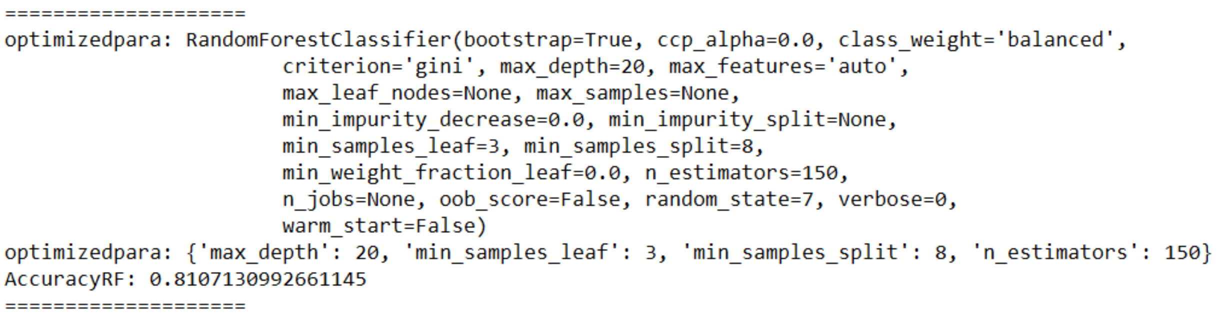
```
====================
optimizedpara: DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=5, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=2, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=0, splitter='best')
optimizedpara: {'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 2}
AccuracyDT: 0.712681031601739
====================
```

Decision tree showed an accuracy of 71%



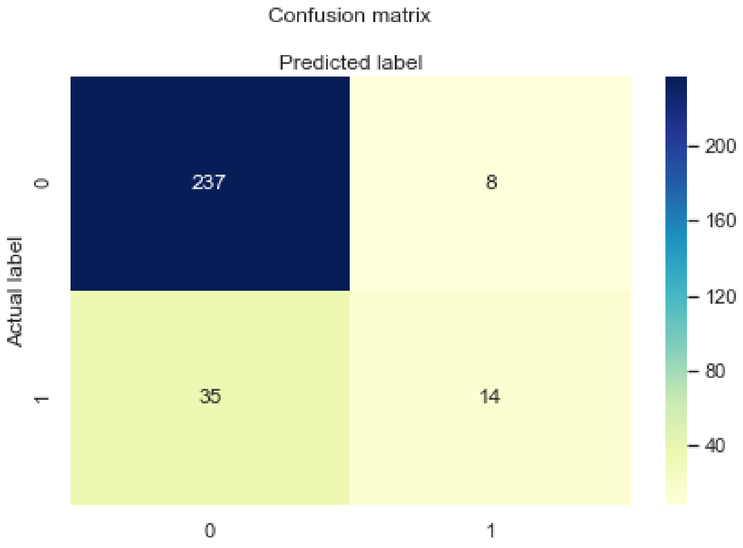
The confusion matrix tells us that the random forest had 232+ 11 predictions correctly and 13 + 38 predictions incorrectly. Leading to the accuracy we got above; 71%

Random Forest:



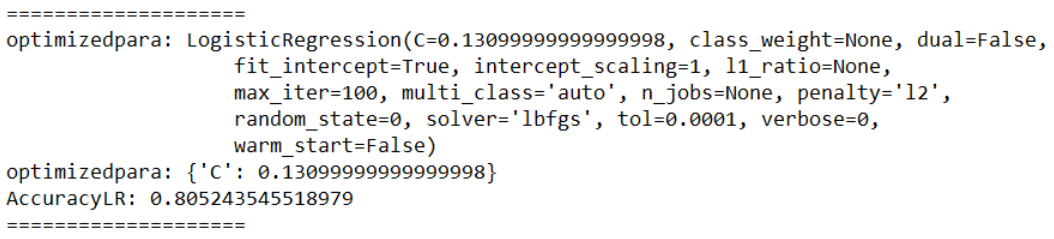
```
====================
optimizedpara: RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight='balanced',
                       criterion='gini', max_depth=20, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=3, min_samples_split=8,
                       min_weight_fraction_leaf=0.0, n_estimators=150,
                       n_jobs=None, oob_score=False, random_state=7, verbose=0,
                       warm_start=False)
optimizedpara: {'max_depth': 20, 'min_samples_leaf': 3, 'min_samples_split': 8, 'n_estimators': 150}
AccuracyRF: 0.8107130992661145
====================
```

The random forest had an accuracy or best score of 81%



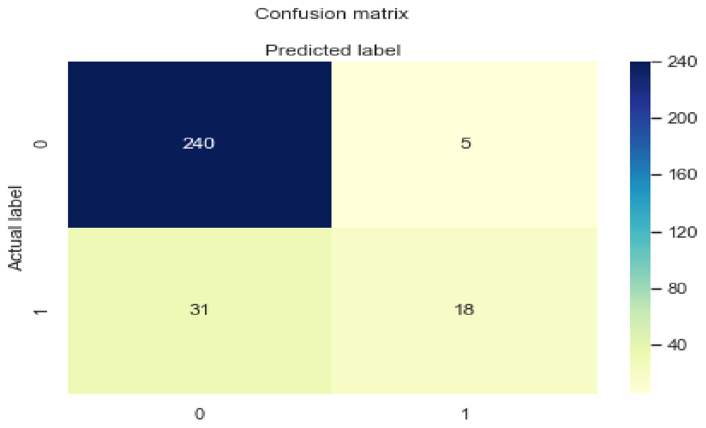
The confusion matrix tells us that the random forest had 237 + 14 predictions correctly and 8 + 35 predictions incorrectly.

Logistic Regression:



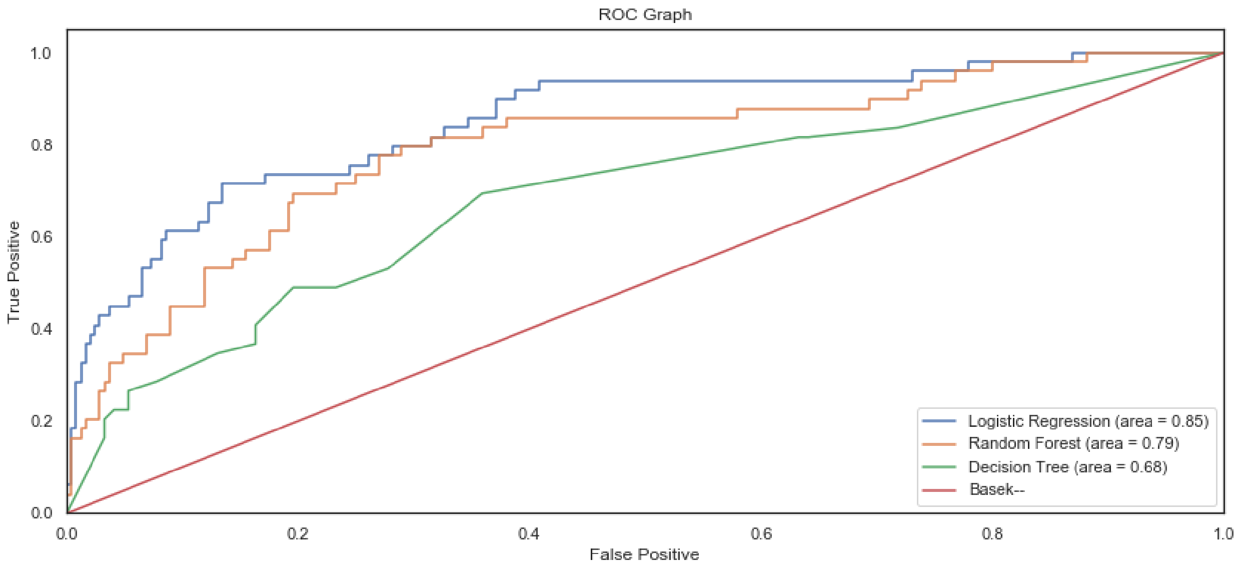
```
====================
optimizedpara: LogisticRegression(C=0.13099999999999998, class_weight=None, dual=False,
                   fit_intercept=True, intercept_scaling=1, l1_ratio=None,
                   max_iter=100, multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=0, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
optimizedpara: {'C': 0.13099999999999998}
AccuracyLR: 0.805243545518979
====================
```

Logistic regression showed an accuracy of 80%



The confusion matrix tells us that the random forest had 240 + 18 predictions correctly and 5+ 31 predictions incorrectly

We plot the ROC curve,



From our analysis, we see that logistic Regression performed the best of all three models having an AUC score of 0.85.